**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 2

**На тему:** *“* Документування етапів проектування та кодування програми”

**З дисципліни:** *“Вступ до інженерії програмного забезпечення”*

**Лектор:**

доц. каф. ПЗ

Левус Є.В.

**Виконав:**

ст. гр. ПЗ-15

Фостяк А.І.

**Прийняв:**

асист. каф. ПЗ

Самбір А.А.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2020 р.

∑= \_\_\_\_ .

Львів – 2020

**Тема роботи:** Документування етапів проектування та кодування програми.

**Мета роботи:** Навчитися документувати основні результати етапів проектування та кодування найпростіших програм.

**Теоретичні відомості**

8) Порівняти переваги/недоліки у застосуванні масивів чи однозв’язних списків для власної програми.

Переваги: можна використовувати одночасно декілька елементів одного типу,фіксований набір елементів,кожен елемент має унікальний індекс,дуже просто доступатися до кожного елемента масиву,не затрачається багато часу.

Недоліки: можливі помилки в результаті виконання програми через некоректну роботу дій над масивом чи з масивом,що спричинить додаткову затрату часу на вирішення цієї проблеми.

23)Які правила запису назв функцій?Навести п’ять прикладів.

* Якщо функція виконує якусь дію ,то варто її назвати відповідно до дії CheckForErrors()
* Якщо виконують певну дію над об’єктом то назву потрібно формувати відповідно до дії та об’єкта виконання дієслово + іменник СalculateInvoice()
* Використовуються префікси для функцій: IsVisible(),IsHitRetryLimit().
* Іноді допомогає вживання суфіксів: Max,Min,..

27)Яка послідовність методів у кожній секції класу у мові С++?Навести приклад.

Рекомендується наступна послідовність методів у кожній секції класу:

* Статичні
* Конструктори
* Деструктори
* Змінні-члени
* Оператори
* Методи-члени

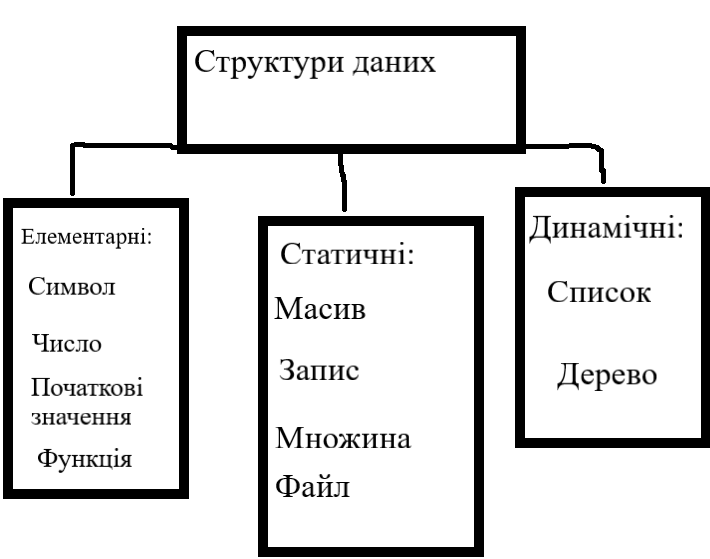
**Постановка завдання**

1.У розробленій раніше програмі до лабораторної роботи з дисципліни “Основи програмування” внести зміни – привести її до модульної структури,де модуль – окрема функція-підпрограма.У якості таких функцій запрограмувати алгоритми зчитування та запису у файл,сортування,пошуку,редагування,видалення елементів та решта функцій згідно варіанту.

2.Сформувати пакет документів до розробленої раніше власної програми:

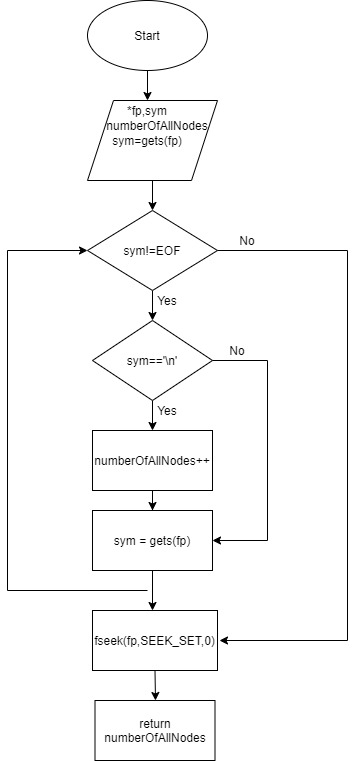
* Схематичне зображення структур даних,які використовуються для збереження інформації.
* Блок-схема алгоритмів - основної функції й двох окремих функцій-підпрограм (наприклад,сортування та редагування).
* Текст програми з коментарями та оформлений згідно вищезгаданих рекомендацій щодо забезпеченості читабельності й зрозумілості.

**Пакет документів**

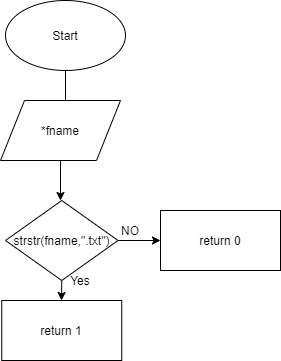
****

На рисунку 1(кінець минулої сторінки) представлено **структуру даних**,що використовуються для збереження інформації.

**Блок-схеми:**

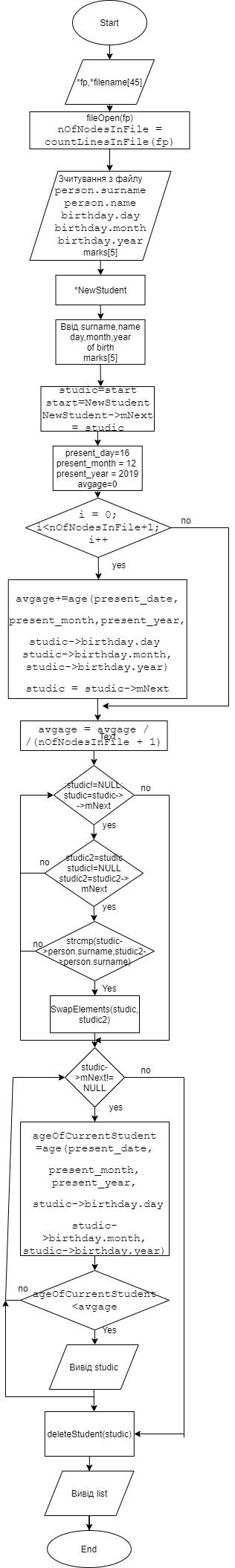


Блок-смеха 1.функції-підпрограми сountLinesInFile



Блок-схема 2.функції-підпрограми testFileName

**Блок-схема3 функція main.c**



**Код програми:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int testFileName(char\* fname);

int countLinesInFile(FILE\* fp);

double age(int present\_date, int present\_month, int present\_year, int birth\_date, int birth\_month, int birth\_year);

typedef struct {

char name[50];

char surname[50];

}Personal;

typedef struct {

int day;

int month;

int year;

}dateofbirth;

typedef struct {

Personal person;

dateofbirth birthday;

int marks[10];

struct student\* mNext;

}student;

void SwapElements(student\* man1, student\* man2);

void deleteStudent(student\* man1);

int main()

{

FILE\* fp;

fp = NULL;

char\* filename[45];

do {

strcpy(filename, "C:\\Andriy\\Cprogram\\lab10\\laba10.txt");

if (testFileName(filename)) {

fp = NULL;

fp = fopen(filename, "r+");//file opening

if (!fp) {

printf("Can't open file\n");

getchar();

}

else {

break;

}

}

else {

printf("File isn't exists\n");

getchar();

}

} while (1);

//check number of lines in file for reading data

int nOfNodesInFile = countLinesInFile(fp);

printf("Amount of nodes in file %d.\n", nOfNodesInFile);

//memory allocating

student\* start = NULL;

student\* studic = (student\*)malloc(sizeof(student));

student\* students = (student\*)malloc(nOfNodesInFile \* sizeof(student));

//cycle for data reading

for (int i = 0; i < nOfNodesInFile; i++) {

if (!start) start = studic;

fscanf(fp, "%s", &studic->person.surname);

fscanf(fp, "%s", &studic->person.name);

fscanf(fp, "%d", &studic->birthday.day);

fscanf(fp, "%d", &studic->birthday.month);

fscanf(fp, "%d", &studic->birthday.year);

for (int j = 0; j < 5; j++) {

fscanf(fp, "%d", &studic->marks[j]);

}

struct student\* past = (student\*)malloc(sizeof(student));

studic->mNext = past;

studic = past;

}

studic->mNext = NULL;

studic = start;

//printing of list

while (studic->mNext != NULL)

{

printf("-------------------------------------------------------------------\n");

printf("|%s %s %d %d %d ", studic->person.surname, studic->person.name, studic->birthday.day, studic->birthday.month, studic->birthday.year);

for (int j = 0; j < 5; j++) {

printf("%d ", studic->marks[j]);

}

printf("|\n");

studic = studic->mNext;

}

printf("-------------------------------------------------------------------\n");

printf("\n-------------------------------------------------------------------\n");

//allocate memory for extra student

student\* NewStudent = (student\*)malloc(sizeof(student));

//scanf data from keyboard

printf("Enter surname of new student:");

scanf("%s", &NewStudent->person.surname);

printf("Enter name of new student:");

scanf("%s", &NewStudent->person.name);

printf("Enter day of birthday of new student:");

scanf("%d", &NewStudent->birthday.day);

printf("Enter month of birthday of new student:");

scanf("%d", &NewStudent->birthday.month);

printf("Enter year of birthday of new student:");

scanf("%d", &NewStudent->birthday.year);

printf("Enter all marks of new student:");

for (int j = 0; j < 5; j++) {

scanf("%d ", &NewStudent->marks[j]);

}

//work with list

studic = start;

start = NewStudent;

NewStudent->mNext = studic;

while (studic->mNext != NULL)

{

studic = studic->mNext;

}

studic->mNext = NULL;

studic = start;

//printing of list with new student

do

{

printf("-------------------------------------------------------------------\n");

printf("|%s %s %d %d %d ", studic->person.surname, studic->person.name, studic->birthday.day, studic->birthday.month, studic->birthday.year);

for (int j = 0; j < 5; j++) {

printf("%d ", studic->marks[j]);

}

printf("|\n");

studic = studic->mNext;

} while (studic->mNext != NULL);

printf("-------------------------------------------------------------------\n");

printf("\n-------------------------------------------------------------------\n");

//current data

int present\_date = 16;

int present\_month = 12;

int present\_year = 2019;

double avgage = 0;

studic = start;

//looking for avgage

for (int i = 0; i < nOfNodesInFile + 1; i++) {

avgage += age(present\_date, present\_month, present\_year, studic->birthday.day

, studic->birthday.month, studic->birthday.year);

studic = studic->mNext;

}

avgage = avgage / (nOfNodesInFile + 1);

printf("Student avg age is : %lf\n", avgage);

studic = start;

//additional memory allocating for swap function

student\* studic2 = (student\*)malloc(sizeof(student));

studic2 = studic->mNext;

//swap elements by surname

for (; studic != NULL; studic = studic->mNext) {

for (studic2 = studic; studic2 != NULL; studic2 = studic2->mNext) {

if (strcmp(studic->person.surname, studic2->person.surname) > 0) {

SwapElements(studic, studic2);

}

}

}

double ageOfCurrentStudent = 0;

studic = start;

//printing students with age less then avgage

while (studic->mNext != NULL)

{

ageOfCurrentStudent = age(present\_date, present\_month, present\_year, studic->birthday.day

, studic->birthday.month, studic->birthday.year);

if (ageOfCurrentStudent < avgage) {

printf("-------------------------------------------------------------------\n");

printf("|%s %s %d %d %d ", studic->person.surname, studic->person.name, studic->birthday.day, studic->birthday.month, studic->birthday.year);

for (int j = 0; j < 5; j++) {

printf("%d ", studic->marks[j]);

}

printf("|\n");

}

else {

}

studic = studic->mNext;

}

studic = start;

//student delete

deleteStudent(studic);

printf("\n-------------------------------------------------------------------\n");

printf("List after delete.\n");

studic = start;

do

{

printf("-------------------------------------------------------------------\n");

printf("|%s %s %d %d %d ", studic->person.surname, studic->person.name, studic->birthday.day, studic->birthday.month, studic->birthday.year);

for (int j = 0; j < 5; j++) {

printf("%d ", studic->marks[j]);

}

printf("|\n");

studic = studic->mNext;

} while (studic->mNext != NULL);

printf("-------------------------------------------------------------------\n");

return 0;

}

int testFileName(char\* fname) {

if (strstr(fname, ".txt")) {

return 1;

}

else {

return 0;

}

}

int countLinesInFile(FILE\* fp) {

char sym;

int numberOfAllNodes = 0;

sym = getc(fp);

while (sym != EOF) {

if (sym == '\n') {

numberOfAllNodes++;

}

sym = getc(fp);

}

fseek(fp, SEEK\_SET, 0);

return numberOfAllNodes;

}

double age(int present\_date, int present\_month, int present\_year, int birth\_date, int birth\_month, int birth\_year)

{

int month[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

if (birth\_date > present\_date) {

present\_date = present\_date + month[birth\_month - 1];

present\_month = present\_month - 1;

}

if (birth\_month > present\_month) {

present\_year = present\_year - 1;

present\_month = present\_month + 12;

}

int final\_date = present\_date - birth\_date;

int final\_month = present\_month - birth\_month;

int final\_year = present\_year - birth\_year;

double final\_age = final\_date + (final\_month \* 30) + (final\_year \* 365);

return final\_age;

}

void SwapElements(student\* man1, student\* man2) {

student man;

memcpy(man.person.name, man1->person.name, 50);

memcpy(man.person.surname, man1->person.surname, 50);

memcpy(man1->person.name, man2->person.name, 50);

memcpy(man1->person.surname, man2->person.surname, 50);

memcpy(man2->person.name, man.person.name, 50);

memcpy(man2->person.surname, man.person.surname, 50);

man.birthday.day = man1->birthday.day;

man.birthday.month = man1->birthday.month;

man.birthday.year = man1->birthday.year;

man1->birthday.day = man2->birthday.day;

man1->birthday.month = man2->birthday.month;

man1->birthday.year = man2->birthday.year;

man2->birthday.day = man.birthday.day;

man2->birthday.month = man.birthday.month;

man2->birthday.year = man.birthday.year;

for (int i = 0; i < 5; i++)

{

man.marks[i] = man1->marks[i];

man1->marks[i] = man2->marks[i];

man2->marks[i] = man.marks[i];

}

}

int check(int arr[])

{

int c = 0;

for (int i = 0; i < 5; i++)

{

if (arr[i] == 4 || arr[i] == 5) { c++; return 1; }

}

return 0;

}

void deleteStudent(student\* cur) {

while (1) {

student\* prev = cur;

cur = cur->mNext;

if (!cur) break;

int a = check(cur->marks);

if (a == 0) {

if (!cur->mNext) {

prev->mNext = NULL;

break;

}

prev->mNext = cur->mNext;

free(cur);

cur = prev;

if (!cur->mNext) break;

}

}

};

**Висновки**

У процесі виконання цієї лаборотарної роботи я навчився документувати основні результати етапів проектування та кодування найпростіших програм.